(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 May 2001 (10.05.2001)

PCT

(10) International Publication Number
WO 01/33356 A1

(51) International Patent Classification[7]: G06F 11/00, 9/44, 17/60, 17/00, 9/45, 17/30

(21) International Application Number: PCT/US00/41894

(22) International Filing Date: 3 November 2000 (03.11.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/163,477 3 November 1999 (03.11.1999) US

(71) Applicant: ANDERSEN CONSULTING L.L.P. [US/US]; 100 South Wacker Drive, Chicago, IL 60603 (US).

(72) Inventor: NICHOLS, David, L.; 4955 South Washington Park Court, Chicago, IL 60615 (US).

(74) Agent: OKEY, David, W.; Brinks Hofer Gilson & Lione, P.O. Box 10087, Chicago, IL 60610 (US).

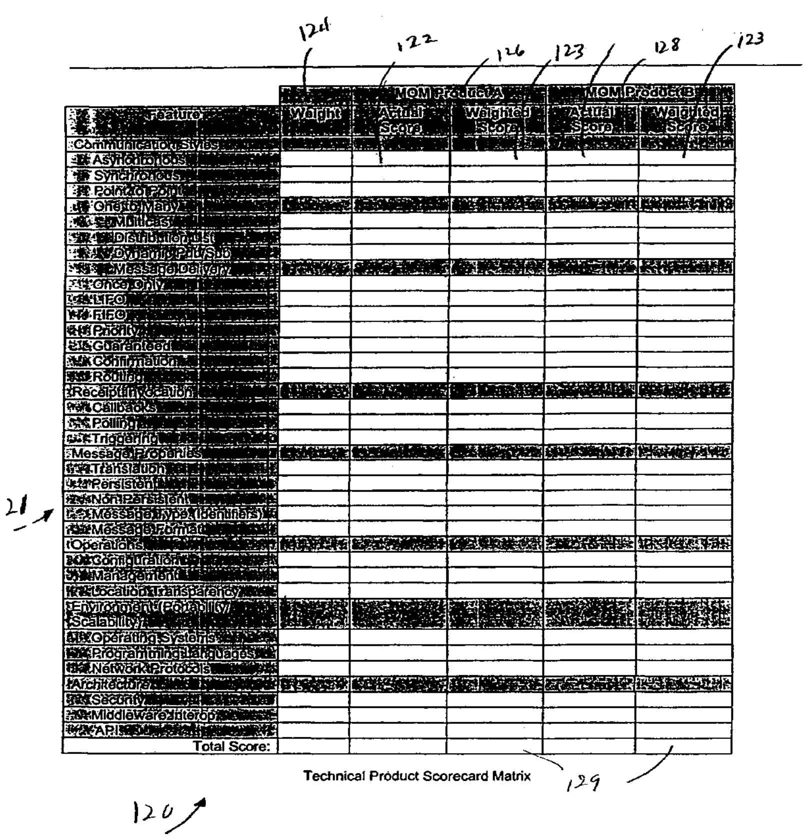
(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

[Continued on next page]

(54) Title: METHOD FOR EVALUATING AND SELECTING MIDDLEWARE



Technical Product Scorecard Matrix

(57) Abstract: A system and method for evaluating and then selecting middleware for computing systems is revealed. The method uses a computer-generated scorecard for producing weighted evaluations of middleware products. The scorecards may include both technical features and non-technical factors in the evaluations.

WO 01/33356 A1

— *Before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD FOR EVALUATING AND SELECTING MIDDLEWARE

REFERENCE TO EARLIER FILED APPLICATION

The present application claims the benefit of U.S. Provisional Application No. 60/163,477, filed November 3, 1999, which is incorporated by reference herein.

FIELD OF THE INVENTION

This invention pertains to software and a method for evaluating software for use in businesses or enterprises.

BACKGROUND OF THE INVENTION

While there are many possible definitions for middleware, one useful definition is that middleware is a set of software services, either custom developed or vendor provided, which enables elements of distributed business or enterprise applications to interoperate. These elements share function, content, and communications across heterogeneous computing environments. Middleware allows for reconciliation of differences and incompatibilities in underlying communications protocols, system architectures, operating systems, databases, and other application services.

Another way to say this is that middleware provides an interface between two otherwise non-communicating systems or services. Middleware typically provides an Application Programming Interface (API) which offers a set of architecture services such as platform and service location transparency, transaction management, basic messaging between two points, and guaranteed message delivery. In particular, middleware hides many of

the underlying technical complexities and difficulties of platforms, databases, and networks.

Middleware is the "glue" that binds business applications to technical infrastructure through programming interfaces. One oversimplified, but vivid way of thinking about middleware, is to consider it as a "transfer protocol", or translator between programs, systems or networks of computers. Middleware links architecture components to deliver more comprehensive services to applications. Middleware enables distributed communications in heterogeneous environments, bridges differences and incompatibilities, and provides architecture services. Figure 1 illustrates the relationship of middleware to business applications and low-level technical infrastructure services.

Middleware provides several key architectural benefits, including providing enhanced architecture functionality and scalability. Middleware enables portability and interoperability, and provides platform and service location transparency. Middleware simplifies programming interfaces, provides adaptability and flexibility, and enforces architecture & application uniformity. Finally, middleware enables de-coupling of application logic, and not the least of the advantages, middleware isolates developers of software, systems, and networks from technical complexity.

Middleware has become so useful and popular that there are now many, many middleware options available to users. The problem, therefore, is not to create middleware, but to determine which middleware may be "right" for a particular application. What is needed is a way to evaluate and rate middleware for a given application. What is needed is a template or protocol for evaluating middleware programs for their ability to perform functions in a way most advantageous and transparent to users.

SUMMARY OF THE INVENTION

One method of practicing the invention assists a user in obtaining middleware software for a computer application. The method includes a step of gathering information about middleware products and their performance. With the information at hand, a user may proceed to evaluate the middleware

products, typically by evaluating or rating a plurality of features of the product or products. After rating the products, the user selects a middleware product for the application. In one method of practicing the invention, a user may gather information about middleware or interface products likely for the application. The user may proceed to evaluate only those products, primarily for the features that are important for the given application, and then select a middleware product.

In another embodiment of the invention, a computer and a computer-generated scorecard is used to rate or evaluate the middleware products available. A computer processor or microprocessor and at least one computer memory is used to generate and store the scorecard. This scorecard may also be in the form of a computer-generated template or macro. A keyboard or computer files may be used to input the ratings or evaluations of the middleware. An output file printer or a CRT monitor may be used to display the scorecard. Such scorecards or scores may tend to make the process more objective and thus easier for a user. Thus, the invention may also include a computer system for practicing the invention.

The invention is further described below, by means of the following figures, which are meant to describe, but not limit, the claimed invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 depicts how middleware fits into computer applications.

Fig. 2 is a representation of native database access in a database access management (DBAM) application.

Fig. 3 represents pathways for another form of DBAM.

Figs. 4a and 4b illustrate database gateway processing in database access middleware.

Fig. 5 is a flow chart for message passing.

Fig. 6 is a flow chart for message queuing

Fig. 7 is a flow chart for a publish and subscribe system.

Fig. 8 is a chart for evaluating middleware.

Fig. 9 is a representation of a middleware system for an Object Request Broker (ORB).

Fig. 10 is a flow chart for a remote procedure call (RPC).

Fig. 11 is a flow chart for a transaction processing monitor (TPM).

Fig. 12 is a scorecard for evaluating middleware technical functions.

Fig. 13 is a scorecard for evaluating non-technical middleware functions.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The invention is a method and computer system for evaluating software for a given application to determine whether a particular middleware software product is suitable for the application, or if several candidates are available, to rate each candidate middleware to determine which is the best fit for the application. In one embodiment, a user determines whether there is a need for a link or application-level application programming interface (API) between two computer products. These computer products may include computer systems, computer networks, operating systems, system architectures, protocols, computer platforms, databases, or other products. If the user determines there is a need for an interface or middleware product, the products available are evaluated, and if one product is suitable for the application, it is selected from among those evaluated. Before describing the various embodiments of the invention, the reader will benefit from the detailed description of middleware products.

Middleware may be considered an interface between a computer application and the technical structure supporting that application. Fig. 1 depicts a business application 10 that interacts with a technical infrastructure 14 through middleware 12. The technical infrastructure may have an application programmer interface (API) 15, and the middleware 12 may also have an API 13 for ease of use. The middleware enables a user to more readily use the technical application, whether it is a platform application, a database, or a technical protocol, or other technical infrastructure. Information technology industry experts and research organizations commonly refer to five major middleware categories: database access, message oriented, object request broker, remote procedure call, and transaction processing monitor middleware. The next section provides definitions for each category of middleware.

Definitions of the Five Major Middleware Categories

Database Access Middleware (DBAM). Database Access Middleware enables a program or process to interact with databases across disparate networks, protocols, and interfaces. There are three types of database access middleware, Client-side database APIs, server-side database gateways, and Native database API. Native database API also includes Embedded SQL (Structured Query Language), Stored Procedure Call, and Multidimensional and non-relational.

Message Oriented Middleware (MOM). Message Oriented Middleware refers to the process of distributing data and control through the exchange of records known as messages. Programs or processes communicate either asynchronously (connectionless) or synchronously (connection-oriented). There are three communication models supported, including message passing, message queuing, and publish & subscribe. This category includes message broker(MB), which differs slightly from traditional MOM, in the use of an intelligent third party to move messages, rather than directly passing them from one party to another.

Object Request Broker (ORB). Object Request Brokers enable program objects and components to access remote objects and components over the network and invoke operations (i.e., functions, methods). ORBs typically provide interoperability between homogeneous or heterogeneous distributed applications; across languages and platforms. Two fundamental types of ORBs exist, COM/DCOM-based (distributed common object model) and CORBA-based (common object request broker architecture).

Remote Procedure Call (RPC). Remote Procedure Call middleware enables a program or process to make calls that are executed in another program or process. The process can be on the same computer or on a different computer in the network. The calls or remote functions are made through an Application Programming Interface (API) as if these functions were called and processed locally, using a function/procedure-calling paradigm. RPCs enforce a common message structure.

Transaction Processing Monitor (TPM). Transaction Processing Monitors provide synchronous messaging and queuing along with other transaction management services designed to support the efficient processing of high volumes of transactions. Core services include load balancing, rollback/commit, and recovery. TPMs also act as a database connection concentrator since programs and processes connect to the TP Monitor and not the DBMS directly.

Descriptions of the Five Major Categories

1. Database Access Middleware (DBAM) enables applications to connect with various databases. This category of middleware typically performs one or more of the following three primary functions. DBAM middleware establishes communications links with server (host) databases, mapping connection paths, establishing links, handling protocol conversion, and delivering data to local points. Secondly, DBAM middleware obtains information about the type and structure of the source database and translates the query accordingly. Thirdly, DBAM translates results of the query into the format that the requesting application is expecting.

Just as there are three primary function of DBAM middleware, there are three forms of Database Access Middleware: native, client-side database APIs, and server gateways. Another option, while not directly considered middleware, is database replication, if requirements are strictly for data sharing. Replication is usually database specific and will not be covered here. Fig. 2 illustrates pathways for native database access. Fig. 3 illustrates pathways for client-side database API's, and Figs. 4a and 4b illustrate pathways in database gateways in two different applications or situations.

In native database access, database vendors incorporate proprietary interfaces to their systems in order to lock customers into their products. Interfacing to a database therefore requires that applications make use of middleware which is native to the database. Use of this native database access middleware is not a choice, but is required. Native database access is most commonly, but does not need be, SQL-based. Multidimensional and/or non-relational data can also be accessed through native database access

solutions. Native database access middleware performs machine translations (e.g., ASCII to/from EBCDIC) and hides network protocols so that the application developer doesn't have to deal with the complexity. Fig. 2 depicts an example of a database program 20 interfacing with a proprietary SQL-net interface 22 between the database and the application 24. An example would be an Oracle database.

An example of a client-side database API is Microsoft's Open Database Connectivity (ODBC). ODBC has become a de facto standard for database independence and the ability to access multiple data sources from a single application. It is the closest the industry comes to a standard database API. ODBC is based on the X/Open-SQL Access Group (SAG) Call Level Interface (CLI) specification. ODBC's purpose is to provide a common, client-side API for accessing data in heterogeneous, SQL-based data managers. Database drivers translate ODBC client APIs into the native equivalent on the back-end database platform. It is important to note that ODBC implementations are limited to two-tier architectures.

Fig. 3 depicts the ODBC architecture. An application 32 uses the ODBC 34 to send SQL statements to a common client-side API 36 for accessing the database 38, and to retrieve the results from the database. Internal ODBC architecture may include a driver manager and one or more ODBC drivers. A driver manager loads ODBC drivers on the application's behalf. Each driver is a dynamic link library (DLL) that processes ODBC function calls (translating them as necessary to the specific syntax acceptable to the database server), submits SQL requests to the DBMS, and returns results to the application. An emerging trend in client-side database APIs is Java Database Connectivity (JDBC).

Organizations are increasingly looking to database gateways as middleware to solve data interoperability. Companies today store data on a variety of platforms, namely mainframes, workgroup and departmental servers, and desktops. Ideally, companies would like to standardize on one platform and data storage mechanism. However, standardization is hardly a realistic option for most companies, given their decentralized organizational

structure and the amount of data currently in legacy systems (DB2, VSAM, IMS, and other data sources). A better option is to rationalize existing databases behind a standard interface and integration mechanism provided by a database gateway.

Database gateway solutions transparently connect a client application to multiple data sources. A database gateway server adds features such as a global data dictionary or catalog for location transparency, heterogeneous distributed SQL (joins and distributed transactions through two-phase commit), and a distributed optimizer. Database gateways are depicted in Figs. 4a and 4b, two configurations being possible with database gateway servers. The first places the gateway between the application and the databases. The second has the application interfacing to a single database and the gateway server connecting the databases together. Regardless of the configuration, the role of the database gateway server is to enable a single application or tool to access multiple, heterogeneous data sources that may be relational and/or non-relational.

Native database access products may include, but are not limited to, Gupta SQL Windows, Intersolv SequeLink, Microsoft SQL Server, Oracle Net8 (formerly SQL*Net), Sybase Open Client/Open Server, and IBM Distributed Database Connectivity Services (DDCS). Client-side database API products include, but are not limited to, Microsoft ODBC, Visigenic Software ODBC Driver, Tandem Computer ODBC Server, and Sun JDBC.

Gateway products may include, but are not limited to, Sybase MDI Gateway and OmniConnect, IBM EDA/SQL Server and Hub Server, Oracle Open Gateway, and IBM DataJoiner. Replication products may be used to identically reproduce data in another form. Replication products may include, but are not limited to Oracle Advanced Replication Option, Sybase Replication Server, ETI Extract, Praxis International OmniReplicator, Platinum Technology InfoPump, and Prism Data Warehouse Manager.

The more common database access architecture, depicted in Fig. 4a, has no gateways between the data requester (the client) and the data provider (the server). An application 41 has an API interface 42 directly to

database 45. The interface may be a native database driver or ODBC middleware. A gateway server 44 then provides an interface between database 43 and database 45. A direct point-to-point connection between the client and the database is established, which can be accomplished using either native database drivers or ODBC middleware. Generally speaking, native database middleware can yield better performance than ODBC or three-tier database gateways.

The less common architecture is the three-tier database gateway depicted in Fig. 4b. An application 41 interfaces through a gateway 44 to one or more APIs, typically on different platforms. The APIs then interface directly with databases 43, 45. A database gateway implies a three-tier architecture where the client, the gateway and the target database(s) are run on different platforms. Using the gateway architecture, each request that is sent to the gateway is parsed, analyzed, optimized and ultimately translated to the target dialect understood by the target database. Gateways are one of the more difficult solutions to implement. They are cumbersome to work with in many cases, and can greatly impact performance. Native drivers are the easiest to use, but limit access to multiple data sources. ODBC helps to alleviate that, but requires a user to merge the data.

2. Message Oriented Middleware. Message-oriented middleware (MOM) is a category of middleware whose purpose is to move a message from one program to another program, generally on another computer. MOM differs from other forms of program-to-program middleware, such as RPCs, because MOM communication is often connectionless, i.e., the sending and receiving programs do not interact directly. Applications that communicate with one another in a connectionless manner are sometimes called decoupled if changes to one application do not directly impact the other.

A program sends the message to the MOM, which then takes responsibility for delivering it to the proper receiver(s). Because of the fact that messages occur between applications, it is possible that there is no need to change the applications to take advantage of messaging systems. In general, MOM lends itself to being less intrusive to application integration

than using an RPC. Message oriented middleware products generally fit into three types: message passing, message queuing, and publish & subscribe.

Message Passing

Message passing middleware is similar to RPCs in that the preliminary aim is to send messages from one application to another. Remote Procedure calls (RPC), which are basically structured sockets APIs, require that the messaging be synchronous and therefore blocking. Message passing middleware, however, allows for synchronous but also non-blocking asynchronous messaging. MOM distributes interprocess communications (IPC) services (e.g., pipes, shared memory, and semaphores) over the network, allowing less structure and more flexibility to the developer. The important end result is that unlike messaging with RPCs, application messaging using message passing middleware can be conversational. Non-blocked applications can receive responses to messages sent either through an event-driven mechanism (i.e., interrupt or call-back function) or through polling. Message passing is depicted in Fig. 5, which shows a sender 52 communicating through a network 54 directly to a receiver 56.

Message Queuing (Store & Forward). Message queuing middleware extends the message passing model to allow messages to be sent to a queue for deferred processing with guaranteed delivery. Queuing systems can act as a resource to a transaction monitor and participate in a two-phase commit session. There is a good deal of effort required to implement message queuing software as compared to straight message passing software. Fig. 6 depicts the situation for message queuing, in which a message from a sender 62 goes into a queue 63 for storing before it is sent to a queue for forwarding 65 to a receiver 66.

Publish & Subscribe. Publish & subscribe (P&S) middleware is event driven middleware. P&S provides the functionality to allow applications to publish and subscribe to events. Consumers register an interest in a piece of data by subscribing. The situation for P&S is depicted in Fig. 7. A publisher 72 sends a message or story to a publisher or message broker 74. The subscribers 76 then receive the story in one of several ways. Publishing

events may be completely independent of any subscriptions. Communication is in one direction only, and is often one-to-many. P&S software can broadcast messages by pre-configured criteria such as user type, subject, etc. P&S can also guarantee message delivery. Many P&S products require substantial effort to implement. Often this type of P&S functionality is simply built on top of the message queuing capability provided by MOM products.

The differences between the various classes of message oriented middleware (MOM) are significant. Below is a general overview of the requirements that lead to each MOM class. Message Passing – Message passing is most appropriate for enabling communications between concurrently executing systems requiring very fast response times and low latency. It is an especially good replacement for batch architectures that require real-time processing. Message passing is the MOM class that provides immediate confirmation that a message has been received. Message Queuing – Message queuing is useful in a wide range of messaging contexts. It provides both high performance (although not quite as high as message passing) and reliability in the form of persistent messages and guaranteed delivery. It is also a good solution for concurrently executing applications if performance requirements allow for a slight time delay. Queuing is the class of MOM that allows for disconnected operation; that is, the ability for applications to communicate without requiring all processes to be active at the same time.

Publish/Subscribe – Publish/Subscribe messaging, like message passing, is best used in concurrently executing systems. It provides very high performance message delivery. Publish/Subscribe is particularly useful in situations where there are a large number of receivers interested in the same information (a 1:n relationship between suppliers and consumers of information) The publisher is not aware if anyone is receiving the messages it sends. Publish/Subscribe is the only class of MOM that is inherently event driven. The decision on which MOM class to use is ultimately driven by the nature of the data being transferred, the size of the data, the frequency of the

data, the latency requirements of the data, and whether the communicating nodes are concurrently executing.

One facet of MOM is Message Broker (MB). MB is an emerging middleware architecture trend that has the potential to improve enterprise-wide integration strategies. The concept is to use an intelligent third party – a broker – to pass messages between multiple disparate information sources and interested consumers. A message broker architecture is an approach for integrating heterogeneous applications. A message broker facilitates program-to-program communication between disparate applications and adds value through message format transformation, message warehousing, flow control, message dictionary, administration and monitoring, protocol adapters, interface adapters, and multiple communication models.

Each of these functions are distinct, and are defined as follows:

Message Format Transformation - Transforms messages from the incoming format to different output formats.

Message Warehousing - Temporary storage of messages to be analyzed and retransmitted at a later time.

Flow Control (Workflow) - Tracks and/or organizes multi-step business procedures.

Message Dictionary - Support for a message dictionary to hold metadata descriptions of message formats.

Administration and Monitoring - Manages the broker configuration.

Protocol Adapters - Bridges dissimilar communication protocols for multiple interfaces.

Interface Adapters - Sophisticated transformation, sometimes including message splitting or combining and algorithmic lookups.

Multiple Communication Models - Support for one-way messaging, queuing or publish-and-subscribe

communication models in addition to request/reply communications.

It is estimated that more than one-half of large enterprises will have some form of message broker in production by 2001. The popularity of MB is due its ability to integrate disparate systems over heterogeneous environments. Heterogeneous environments consist of hardware platforms, operating systems, programming languages, object brokers and applications. MB has emerged as the fastest growing kind of middleware, largely because of its versatility. Application designers or integrators can use MB for several implementation styles, including use for disparate environments, changes to accommodate customer expectations, and for transformation of data processing operations that were not designed for modern, high-speed, data communications.

The most common use of MB is by enterprise organizations that have a mix of heterogeneous systems and applications running on disparate environments. They seek to reduce application integration costs by centralizing the implementation, maintenance and management of required interfaces, thus going from point-to-point ("spaghetti") massaging to a well managed application integration (message broker) environment.

Another common use of MB is for re-engineering processes to accommodate customer focused views that require close co-operation between previously incompatible applications. This use may apply both to businesses and also to enterprises not in business, but with communications and computing needs as essential to their operations as any business. This latter may include governmental organizations, non-governmental organizations such as charitable, educational, cultural, civic or other non-profit groups. The utility of the invention is not limited to any particular group or type of group.

The other common use of MB is for routing, transforming and distributing transactions from a Web server to multiple back office applications, applications which were never conceived with the Internet in mind. Some of Message Broker functionality may be purchased, while other

functionality is typically developed to fit a client's environment and requirements. It is recommended that migration to a Message Broker architecture to support the integration of enterprise systems should be implemented gradually.

A number product options are available in the MB category. These options include message distribution programs, message warehouse programs, message transformer programs, message dictionary programs, and interface engines. Interface engines may include, but are not limited to, Century Analysis Transaction Data Manager, Healthcare Communications/Perceptive Cloverleaf, Hublink Integrator, Muscato Engine, and Software Technologies DataGate, and are potentially useful for any of the other purposes mentioned, including acting as a Message Warehouse, Message Transformer, Message Dictionary, or a Message Distribution Mechanism.

Message distribution programs include the interface engines mentioned, and also include, but are not limited to, ACI Net 24, Deluxe Data Architect, Digital Equipment's DECMessageQue, IBM MQSeries, NEON Software NEONet, S2 Network Express and TIBCO Rendezvous. Message Warehouse programs include, but are not limited to, BEA Message Q, IBM MQSeries, TIBCO Rendezvous, EDI Clearinghouse (Electronic Data Interchange), and the interface engines mentioned. Message transformers include, but are not limited to, the interface engines, TIBCO TIB/MessageBroker, Apertus Enterprise Integrator, New Paradigm Copernicus, Sterling Software Gentran, and TSI International Mercator. Message Dictionary programs useful for this purpose include, but are not limited to, Apertus Enterprise Integrator, TSI International Mercator, Early Cloud Message Driven Processor, IBM Flowmark Application Integration Facility, and Momentum Software Visual Flow.

The sheer number of programs available suggests that a user may want to evaluate one or more of these software options before purchasing and using.

To help understand the strengths and weaknesses of each MOM class, Fig. 8 may be used by architects to help determine which features are necessary for a given architecture based on the system requirements. If this does not provide enough information to reach a decision, the architect should weight or rank various features based on their perceived importance. Given well-defined system requirements and a good understanding of MOM, the choice of MOM class should be clear; however, sometimes hybrid solutions may be required.

Fig. 8 presents in diagrammatic form a summary of the features 81 typically present in MOM products of the three classes 83 discussed above. The diagram depicts in "Harvey ball" symbolism whether a feature is present as a primary or secondary feature, or not at all. Those skilled in the art will recognize that Fig. 8 is itself a quick evaluation of middleware for MOM applications.

3. Object Request Brokers. Object Request Brokers (ORBs) are primarily useful for deploying object-oriented, componentized systems that need to operate in a heterogeneous environment. Put another way, ORBs are used to find (business) data in distributed environments. Of course, as applied to a non-business enterprise, the goal of finding information may be the same, but the information sought may be of a different nature. The typical role of an ORB is to enable communication within one application domain with consistent design characteristics. There are two "standards" for implementing distributed component messaging – the Object Management Group's CORBA, an open standard, and Microsoft's DCOM, which although not an open standard, is in use due to Microsoft's dominant position in the marketplace. The two implementations do not readily interoperate. Messaging between CORBA ORBs and DCOM ORBs requires the use of a gateway solution.

CORBA - the Common Object Request Broker Architecture - is an industry standard for ORBs as defined by the Object Management Group (OMG), a consortium of practically all the major software, hardware, and system vendors (IBM, HP, Oracle, & SUN, among others. The two notable

exceptions being Microsoft and Intel.) An OMG ORB is expected to support request dispatch, parameter encoding, message delivery, synchronization, activation, exception handling and security mechanisms. Various Object Services are addressed by different standards (CORBA services), e.g., event notification services, life cycle services, persistence services and concurrency control services. Common Facilities are services which are useful in some application domains, but need not be offered by an ORB implementation. Examples of such services are User Interface facilities, Information Management facilities and System Management facilities. Domain Interfaces are standardized interfaces designed with specific tasks in mind for distinct vertical markets or industries.

DCOM - the Distributed Common Object Model - has become a de facto ORB standard due to Microsoft's dominant market presence. DCOM uses Microsoft's DCE RPC extension to make method calls across a network. DCOM is strictly proprietary to Microsoft platforms, but most major CORBA vendors are beginning to provide bridges that enable communication between their ORBs and DCOM.

Fig. 9 illustrates a middleware system for ORB. A client application 92 seeks to communicate with a remote service 96. The client application locates an ORB 94 to activate the service 96, thereafter establishing communication between the client and the server, and allowing direct communication between the client and the service. Examples of middleware for ORB include Iona Orbix, Microsoft DCOM, Visigenic (from Borland) Visibroker, BEA Object Broker, v. 3.0; and Expersoft PowerBroker.

One of the most used applications of ORBs is to wrap legacy applications. Retrofitting an existing application with an ORB-based architecture - where the same ORB runs on all relevant platforms - allows for simple communication: synchronous request/reply between one requester program and one server program. ORBs on mainframes allow mainframe applications to interact as equal participants with other CORBA-based applications. Placing the ORB on a mid-tier system (e.g., NT or Unix) is more flexible, with no impact on the mainframe. This is the more common solution,

since few ORBs exist for mainframe systems (one example is Iona's Orb on MVS). Integration with a Legacy system is therefore usually some sort of "wrapper" in conjunction with a MOM product.

An important factor when developing ORB applications is to choose the appropriate level of business component granularity. Large-grained components have greater reuse power, while smaller-grained components have better reuse flexibility. By creating smaller components, functionality is increased but maintenance is made harder in terms of scalability and maintainability. The main issue here is network traffic. "True distributed" object systems using ORBs are unfeasible due to significant network overhead.

A newer "class," as they might be called, of Object Request Brokers, is Object Transaction Monitors. An OTM is an application platform, like mainframe CICS in function but goes beyond the most ambitious middleware product. Conceptually, an OTM is a hybrid of TP monitor and ORB technologies, combining the efficiency and security of TP monitors with object-oriented management. At the time of writing, there is no one product on the market that currently qualifies to be a true OTM product. However, a number of products on the market that may be useful include, but are not limited to, IBM TS, Microsoft TS, Oracle NCA, Sun's Object Transaction Monitor, and similar products from BEA, Sybase, Expersoft, Iona, ICL, SNI, Visigenic, ISIS and UniKix.

Traditional transaction monitors manage the complex tasks of synchronizing the information contained in computer databases. Transactions can include everything from a bank deposit to a customer service inquiry. The distributed Internet environment poses new challenges to older transaction systems that are best solved through the flexibility and power of object-oriented technology. OTM is a crucial piece of the infrastructure required to develop robust transaction-based applications for the Internet or intranet. It integrates transaction management services from the database and mainframe worlds with object-oriented technology.

OTM is highly suited for enterprises where some kind of ORB environment exists and needs to be upgraded to include secure transactional

capabilities. OTM is most appropriate for applications that are complex and need to be changed or revised often. OTM provides a component approach, which could provide a way to divide an application into easy to handle modules.

Enterprises will benefit from the rich infrastructure of an OTM without having to manage all its technical detail. OTMs are seen as the next step toward seamless consolidation of disparate applications. Application designers or integrators can implement OTM in the three following environments: (1) transaction oriented environment where ORB's flexibility is required; (2) adding secure transactional capabilities to ORB's environments; and (3) replacing and upgrading exiting ORB's environments.

4. Remote Procedure Calls. Remote procedure calls (RPCs), one of the better known types of middleware, have been available in UNIX networking since the mid-1980's. Developed by Apollo Computer (now owned by Hewlett-Packard), Sun Microsystems, the Open Group's DCE, and others, RPCs shield application developers from network complexities, providing a foundation for interoperability within and between homogeneous and heterogeneous platforms. Using TCP/IP sockets to communicate, RPCs extend the conventional procedure-based programming model to distributed networking in a manner that is similar to standard subroutine calls executing locally.

RPC middleware redirects procedure calls to some other processor for execution and returns procedure results to the caller. RPC APIs that facilitate send and receive functionality are relatively simple and contain only a few verbs (e.g., connect, send, receive, disconnect). RPCs are conceptually simple, as shown in Fig. 10. An RPC application 100 includes a client 101 making a request of a (typically remote) server 104, as described above. The server executes the request and replies to the client 101. Both the request and the reply are typically made through architecture layers 102, 105, and a client stub 103 and a server stub 106.

In general, custom extensions to the architecture layer are required to incorporate services such as error handling, security, and directory. Many

multi-tier applications (e.g., SAP) as well as high level middleware such as ORBs and TPMs are implemented today using RPCs and similar request/reply mechanisms. "RPC" is sometimes used in a general sense to describe any request/reply (or call-and-return) middleware service that provides basic data-type translation and connection-oriented communication services. In other contexts, "RPC" refers just to those products that use an interface definition language (IDL) for describing the argument lists for outgoing and incoming parameters. Interface definitions using IDL are the input for compilers that generate client stubs and server stubs.

RPCs provide the communications foundation for scalable processing domains that can approximate the power of mainframe computers. This allows inexpensive computers to achieve the same effective processing capability and is a prime motivator for some reengineering and downsizing efforts. However, building an architecture using RPCs is a significant custom development effort. In addition, considerable testing is required to achieve an industrial strength solution, and few tools exist today to facilitate architecture development and testing with RPCs.

A proven technology but lacking in service offerings, RPCs are less robust than alternative middleware technologies for supporting mission-critical, client/server applications. It is therefore rare today for architectures to be based solely on RPCs. Only in situations where there was significant custom application development would RPCs be required as the foundation for architecture development. However, many higher level middleware products (e.g., TPM, ORB, and DBAM) use low-level RPC middleware as the foundation for distributed communications across networks. RPCs and similar request/reply services are seldom used as stand-alone architecture solutions; they are often combined with some set of other services. Some products are relatively unbundled. Others are primarily database gateways that incorporate an RPC mechanism as a supplement to their primary mission.

RPCs or RPC-like mechanisms are typically embedded in transaction processing monitors, such as IBM's CICS, Transarc's Encina, and BEA's

Tuxedo. RPCs are also embedded in development and run-time environments, such as Borland/Open Environment Corp.'s (OEC's) Entera, Seer's High Productivity Systems (HPS), and Software AG's Entire. Application packages, such as SAP's R/3 Remote Function Calls also may include an RPC or RPC-like mechanism. Although advanced when compared to coding down to the sockets layer (RPCs add structure by defining inputs and outputs for messaging) RPCs are generally too low-level for common implementations. Most prefer to buy rather than build functionality. Products are available from Microsoft, NobleNet, Sunsoft and Open Group DCE.

5. Transaction Processing Monitors (TPM). The most significant feature of transaction processing monitors is the ability to funnel database requests from a large number of clients. TPMs allow applications to break up complex transactions into smaller units, and then ensure the completion of those transactions. Transaction processing monitors provide a management layer between the client and server to control all transactions that move through the system. A TPM is depicted in Fig. 11. An example of a use for a TPM is to perform an audit. The TPM 110 includes a Resource Manager 112 to interface with a plurality of clients or customers 114. The TPM then uses application logic 116 to break up complex transactions into smaller units. Transaction processing managers or monitors 118 provide a management layer to control all transactions moving through the system. A communication 111 resource manager monitors links between the database 113 and the TPM to ensure efficient communications in the large number of transactions that are occurring.

TPMs are best suited for the high volume mission critical applications where heterogeneous two-phase commit – the ability to synchronize updates across two or more databases from different vendors – is needed. A three tier client/server system that employs a TP monitor can manage database requests of more than 1,000 users using only a minimal number of database server connections. TPMs perform best if the number of services is kept to a

minimum. Similarly, locating processes on the client helps to minimize network traffic.

TPMs are much more than two-phase commit. They provide two other distinct types of services essential to building scalable distributed applications. First, they offer robust and efficient communication services between clients and servers and between one server and another. This gives applications the ability to run efficiently over low-speed, high-latency WANs, including the Internet. Second, they offer server-based execution and state management plumbing that enables user-written services to efficiently handle thousands of client requests.

Furthermore, most TPMs have intelligent routing algorithms which dynamically route transactions to the "best" application server, based on message content of an actual performance information obtained by monitoring the application server. Overloaded or failed application servers are detected, and the workload is automatically routed to a proper server while recovery takes place, significantly improving availability. TPMs offered for mainframe environments include IBM's CICS, TPF and IMS/DC products. Microsoft offers Microsoft Transaction Server (MTS), and other products available for client/server environments include Transaction Server from IBM, TUXEDO from BEA, Transarc's Encina, and NCR Top End.

Scoring

While the above list of applications and middleware is extensive, it is not necessarily exhaustive, and all middleware is within the scope of the present invention. Having now listed possible middleware options, a user proceeds to evaluate one or more of the available middleware products for a given application. A scorecard for evaluating products has been found useful, and may be used in a computerized or a manual version. Scorecards or scores will desirably utilize a "weight" or relative importance of each feature on which a particular middleware is rated or scored. Each feature may then be evaluated using a score for each feature. The scores are then multiplied by the weights and a total score for a middleware product is achieved.

Choosing a vendor and a product that best suits a project's needs can be a difficult decision. It is easier if decision factors are presented clearly. This section describes the use of product scorecards to facilitate the presentation of information. This will allow an architect to visually weigh products against a desired system to determine which product is the best choice. For a given project, some product features are never noticed, some are paramount to the design of the system, while most other features fall somewhere between these extremes. To acknowledge this, each product feature is assigned a weight reflecting its importance to the particular project. Table 1 below lists weights for each feature and a definition corresponding to each "weight."

Table 1.

Weight	Description
0	Feature is not related to this project.
1	Feature may be utilized in the future.
2	Feature may be used minimally.
3	System design is based on this feature.

Once each feature has an associated weight, each vendor's implementation of this feature may be assigned a score, such as a score from 0 to 3. Table 2 lists one method of evaluating individual features.

Table 2

Score	Description
0	This product does not support this feature.
1	The product implementation of this feature is poor.
2	This feature has limited support in this product.
3	The product strongly supports this feature.

In the example of Table 3, the use of the rating system is illustrated for a security feature of three different TPMs.

Table 3

Product	Feature	Score	Comment
Product Z	Security	3	The security feature is easy to use, well documented, and complete. Once a user is authenticated, the authorization level of the user determines their access rights and which resources are available. All communications use 128 bit encryption algorithms.
Product Q	Security	2	The product offers authentication, but each user has the same level of authorization once authenticated. The product allows the user to select an encryption algorithm.
Product Y	Security	1	This product can authenticate the users, has only one level of authorization, and only supports 64 bit encryption.

Once a weighting system is in place, and a scoring system as described above has been propounded, a user is in position to evaluate one or more middleware products. An embodiment of a scorecard or full evaluation sheet is depicted in Fig. 12. In this example, a scorecard 120 lists one or more features 121 along the rows of the scorecard. Weights 124 are given in a separate column, one weight to be assigned for each feature as described above. In this example two middleware MOM products 126, 128 are evaluated, and the scorecard has separate columns 122 for the actual score or rating for each product, and weighted score columns 123 for each product to be evaluated. In practice, a user then rates each feature for its importance in the application and assigns a weight. Each feature is then evaluated for its vendor's implementation of the feature. The evaluation score for each feature is the multiplied by the weight assigned to that feature, and the result is entered in the column for "weighted score" 123. The weighted scores are summed, and entered in the row entitled "total score" 129. The user now has an evaluation for each middleware product evaluated.

Features evaluated may include, but are not limited to, the ones depicted in Fig. 12. While this example featured MOM, other features may be extracted and used in a rating sheet for other types of middleware. The user is not bound to use only these features, even on MOM middleware. If a user

perceives other features that are more useful, the user may select and rate those features, in addition to, or in place of, the ones used in Fig. 12. Of course, it will be recognized that "weights" or weighting factors may add value to this process. In the interest of economy, or if weighting factors are not known, this factor may be converted to "unity" weighting, or 1.0, for each factor. It is preferred, but not required, that weighting factors be used.

In one embodiment, a scorecard for middleware may be devised and stored on a computer system for evaluating middleware. A preferred system has a different scorecard adapted for each type of middleware, i.e. one card for DBAM, one for MOM, one for TPM, etc. Especially in a large organization, or one that faces middleware questions often, such a system will desirably save time and effort on the part of software engineers and project managers. The scorecard may then be accessed and used to evaluate middleware. In another embodiment, engineers or personnel may be assigned to evaluate middleware or to track down ratings of middleware to use in such a system.

One aspect of a method for obtaining middleware includes designing the technical architecture necessary for implementing the middleware. Thus, the designer must implement the middleware in a technical architecture that allows the middleware to function in the way it was intended to function. Compatability and implementation instructions are usually included with middleware products and should be followed unless circumstances allow a user to go outside the bounds prescribed. Many of the technical architectures are discussed in the sections above on specific types of middleware categories.

Another aspect of a method for obtaining middleware is to implement a compatible architecture and to install the middleware on a computer system. Each middleware product and each computer system may be different, but the task is the same: install a compatible architecture, install the middleware, and adapt and use the middleware in the computer system. The discussions above for each type of middleware discuss some of the techniques for installation and implementation.

A computerized system will clearly assist in accessing, storing and sharing information concerning middleware products. Such a computer

system may be any convenient computer or computer system, including, but not limited to, a stand-alone personal computer, a remote-type 3270 terminal, or a networked personal computer. A scorecard may be devised by any convenient means and input into the computer. Convenient means include word processors and available spreadsheets, such as Microsoft's Excel® program, or Lotus 1-2-3 spreadsheet or a Visi-Calc spreadsheet. Such a spreadsheet as depicted in Fig. 12 may easily be devised and stored for ready, repeated use. It may even be made into a macro for repeated use in such a manner that the basic spreadsheet is not permanently altered when used.

A computerized system according to the invention will desirably also have at least one memory operably connected to a processor or microprocessor of a computer. Such a memory may be any computer memory convenient for the application, and may be RAM, ROM, or other types, and may be supplied as a hard drive, a disc drive, or other convenient means. The computer system will desirably have a way to input data into the memory of the computer processor or microprocessor. These input means may include a keyboard, a file, a scanner, a bar code reading device, a microphone, a transducer and digitizer, or other means or combination of means to input data. Alternatively, inputs may be accepted by clicking (as with a mouse) on a graphical user interface. Such graphical user interfaces may include, but are not limited to, radio buttons, pull-down menus, pointers, check boxes, list boxes, drop-down boxes, and the like. The results of the rating may then be displayed to a user by the computer system. Suitable displays may include, but are not limited to, a video output, such as a CRT or flat-panel screen, a printer, a voiced output, a computer file, or other convenient means.

In addition to middleware product features, there are other factors that should be considered when choosing a vendor and product. The product under consideration may be technically exceptional, but if enough of certain non-technical features or items raise issues, another product may be chosen. These features, or lack thereof, can usually be worked around. It is important to understand the impact of these items so that they may be allocated extra

time in a planning phase for a given project. One aspect of any middleware project is to plan a course of action for purchasing, installing and implementing the middleware. Factors to include in the plan include, but are not limited to, documentation for the middleware, cost, ease of installation, ease of configuration, administration costs, product support, vendor reputation, client skill base for the product, and client bias for the vendor. These considerations, while not necessarily merely technical in nature, should also weigh into the purchasing decision.

Fig. 13 therefore depicts a second scorecard for non-technical considerations in evaluating middleware. One or more considerations or factors 131 are used. Weights 134 are assigned according to the importance of the factors. One or more products 136, 138 are evaluated, with raw score columns 132 and weighted score columns. At the conclusion of the ratings, the weighted scores are totaled and entered in the total score boxes 139. This weighting format may be carried out by hand calculations, or preferably in a computer environment, such as the environment noted above for a computer spreadsheet program. The weights may be entered along with formulae for multiplying them by the ratings or evaluations, along with an automatic adding that returns a numerical value as soon as the inputs, that is, the weights and the ratings, are entered.

While weights are preferable, it is recognized that no weights or unity weights may also used. These considerations are not exclusive and others may be used, or certain of the considerations depicted may go unused in certain applications and in certain environments. Documentation consideration may include an evaluation of the availability, completeness, or ease of understanding of the documentation. One consideration may be whether the documentation includes working examples. The cost of the middleware and the licensing scheme or schedule may be an important consideration.

Ease of installation is desirably considered, in light of whether the purchaser has the skill sets necessary for a particular middleware product. Ease of configuration is a consideration when a user considers the effort required to setup new nodes, channels, networks, queues, etc.

Administrative costs, the amount of effort and expense required to maintain the product effectively, should also be considered. Quality and availability of product support, from the middleware vendor or from third parties, should be considered before purchasing. Other considerations may include the reputation of the vendor and the number of installations accomplished. Existing in-house experience with the product should be considered as part of the buyer's skill base for the product. Finally, any bias toward the vendor, such as a preference for a certain platform or product, should at least be recognized.

In one embodiment of the invention, a user considers whether there is a need for middleware in a computer application. The user proceeds to evaluate the middleware products available, using a scorecard such as one shown in Fig. 12, after first selecting weights and criteria or considerations that are relevant or important to the application. The criteria may also include non-technical factors, such as those depicted in Fig. 13. These criteria may also be important, as outlined above, but pertain more to the business, cost, or administrative effort of using a particular middleware product.

While this invention has been shown and described in connection with the preferred embodiments, it is apparent that certain changes and modifications, in addition to those mentioned above may be made from the basic features of this invention. Many types of organizations besides businesses may benefit from the use of this invention, e.g., any organization wishing to use or evaluate middleware. These may include governmental organizations, non-governmental organizations such as charitable, educational, cultural, civic or other non-profit groups. The utility of the invention is not limited to any particular group or type of group. In addition, there are many different types of relevant factors to consider in evaluating middleware. Computer or hand scoring may be utilized in practicing the invention.

The invention is not limited to the examples given above. While it is preferable to evaluate middleware from a variety of viewpoints, and from a variety of calculations, the invention may also work with a more limited set of considerations, that is, the considerations or factors most important to the

application, in the opinion of the user, an information systems department, or a person charged with maintaining a computer system utilizing the middleware. Accordingly, it is the intention of the applicant to protect all variations and modifications within the valid scope of the present invention. It is intended that the invention be defined by the following claims, including all equivalents.

I Claim:

1. A method for obtaining a middleware product for an application, comprising:

gathering information about middleware products;

evaluating middleware products; and

selecting a middleware product for the application.

2. The method of Claim 1, further comprising planning a course of action to implement the application.

3. The method of Claim 2, further comprising designing an architecture to implement the application.

4. The method of Claim 3, further comprising implementing said architecture and installing the selected middleware product on a computer system.

5. The method of Claim 1, further comprising identifying middleware products that are available and selecting at least one middleware product to be evaluated.

6. The method of Claim 1, wherein evaluating middleware products includes using a scorecard or template to evaluate at least one middleware product.

7. The method of Claim 1, wherein evaluating middleware products includes using a scorecard or template on a computer to evaluate at least one middleware product.

8. The method of Claim 6, further comprising calculating a numerical score and displaying it on the scorecard or template, wherein the numerical score represents the evaluation of said middleware product.

9. The method of Claim 6, wherein the numerical score is a sum of weighted ratings, wherein each rating corresponds to a feature of a middleware product.

10. The method of Claim 9, wherein the weighted ratings are produced by multiplying the rating of a middleware product feature by a weight, wherein said weight represents a subjective emphasis assigned to a product feature relative to other product features.

11. The method of Claim 10, wherein said weight is selected from the group consisting of 0, 1, 2 and 3, and wherein 0 means the feature is not related to this project, 1 means the feature may be utilized in the future, 2 means the feature may be used minimally, and 3 means that the software design is based on this feature.

12. The method of Claim 1, wherein the middleware evaluated is database access middleware, message oriented middleware, remote procedure call middleware, object request broker middleware, or transaction processing monitor middleware.

13. The method of Claim 1, wherein the step of evaluating includes rating at least one technical feature of the middleware, and inputting said rating on a scorecard.

14. The method of Claim 1, wherein the step of evaluating includes rating at least one non-technical feature for the middleware, and inputting said rating on a scorecard.

15. The method of Claim 8, wherein the middleware with the highest numerical score is selected.

16. A computer system for evaluating middleware products, comprising:

a computer processor;

at least one memory operably connected to said processor, said memory useful for storing data concerning evaluating middleware products; and

a scorecard for evaluating middleware products, said scorecard stored in said memory;

display means, operably coupled to said processor; and

input means, operably coupled to said processor,

wherein a user may input data into the system through said input means, and the scorecard is displayed on said display means to evaluate middleware products.

17. The system of Claim 16, wherein the scorecard includes at least one rating on a technical feature on at least one of the middleware products.

18. The system of Claim 16, wherein the scorecard includes at least one rating on a non-technical feature of at least one of the middleware products.

19. The system of Claim 16, wherein the computer calculates a weighted rating for at least one feature of a middleware product.

20. The system of Claim 19, wherein the weighted ratings comprise weights selected from the group consisting of 0, 1, 2, and 3, and wherein 0 means the feature is not related to this project, 1 means the feature may be utilized in the future, 2 means the feature may be used minimally, and 3 means that the software design is based on this feature.

21. The system of Claim 16, wherein the middleware product is database access middleware, and the technical features include at least one of access form, replication, communication styles, operations, environment and architecture.

22. The system of Claim 16, where the middleware product is message-oriented middleware, and the technical features include at least one of communication styles, receipt invocation, message properties, operations, environment and architecture.

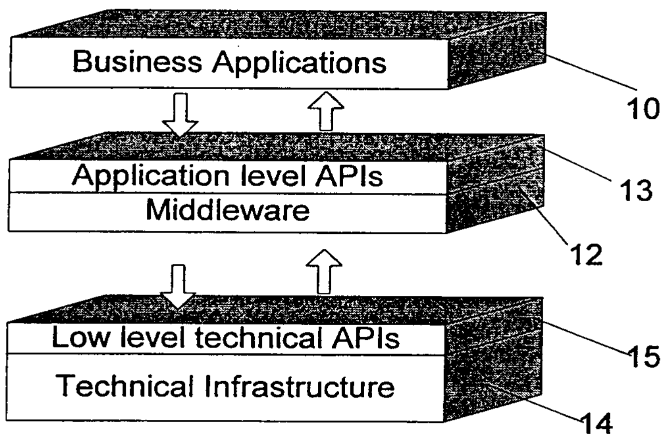
23. The system of Claim 16, where the middleware product is remote procedure call middleware, and the technical features include at least one of communication styles, extension availability and ease of use, receipt invocation, message properties, operations, environment and architecture.

24. The system of Claim 16, where the middleware product is object request broker middleware, and the technical features include at least one of a standard for implementing distributed component messaging, communication styles, receipt invocation, message properties, operations, environment and architecture.

25. The system of Claim 16, where the middleware product is transaction processing monitoring middleware, and the technical features include at least one of routing volume, communication styles, routing algorithms, message properties, operations, environment and architecture.

26. The system of Claim 16, where the non-technical factors include at least one of documentation, cost, ease of installation, ease of configuration, administration costs, product support, vendor reputation, client skill base for the middleware product, and client bias for a vendor.

Fig. 1



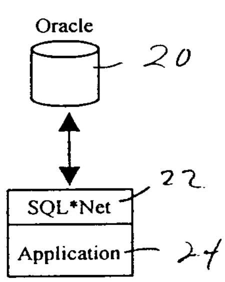


FIG. 2

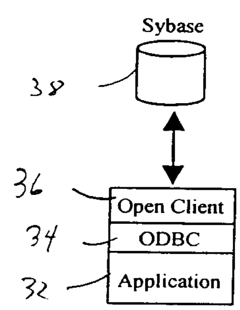
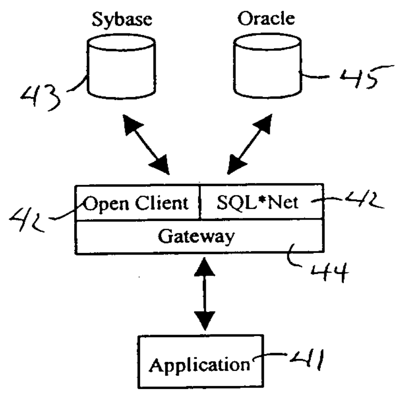
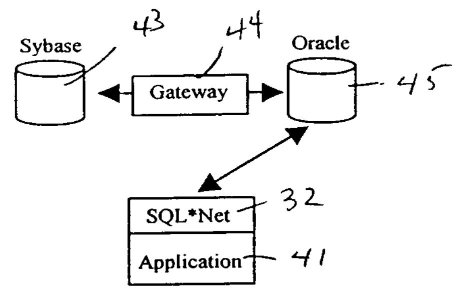


FIG. 3



Database Gateway

Fig 4b



Database Gateway

Fig. 4a

Message Passing

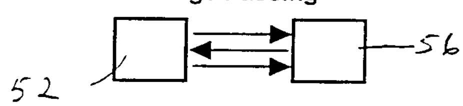


Fig 5

Message Queuing (Store & Forward)

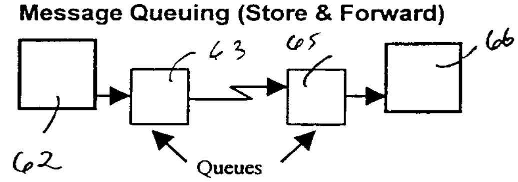


Fig. 6

Fig. 7

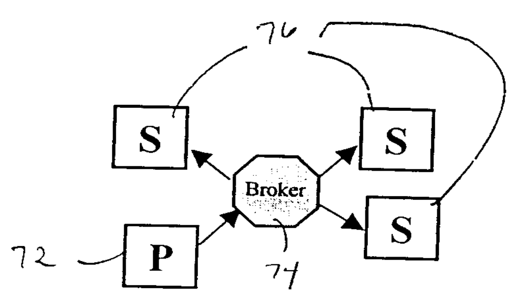


Fig. 8

83

81

Task	Message Passing	Message Queuing	Publish and Subscribe
Sending and receiving applications must be continuously on-line	●	○	●
Sending and receiving applications must be continuously on-line	●	○	○
Support blocking messaging	●	◐	◐
Supports non-block messaging	◐	●	◐
Callbacks can be used	●	○	●
1:n relationships	○	◐	●
Dynamic receivers	○	○	●
Anonymous receivers	○	○	●
Event Programming model	○	○	●
Subject based addressing	○	○	●
Guaranteed delivery	●	●	◐
Persistent	○	●	○
Triggering	○	●	◐

● Primary Functionality
◐ Secondary functionality
○ Not available

Fig. 9

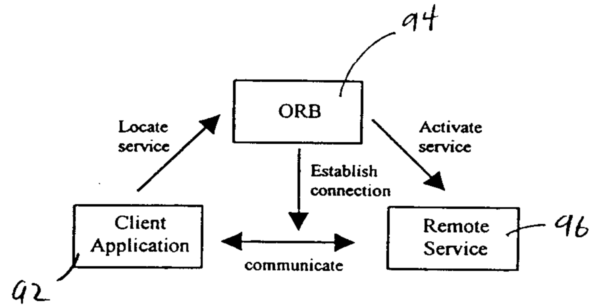


Fig 10

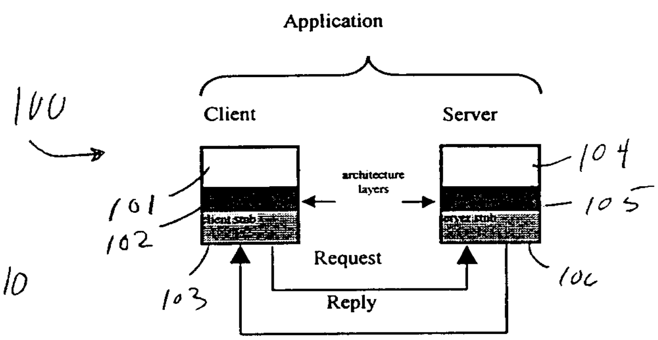
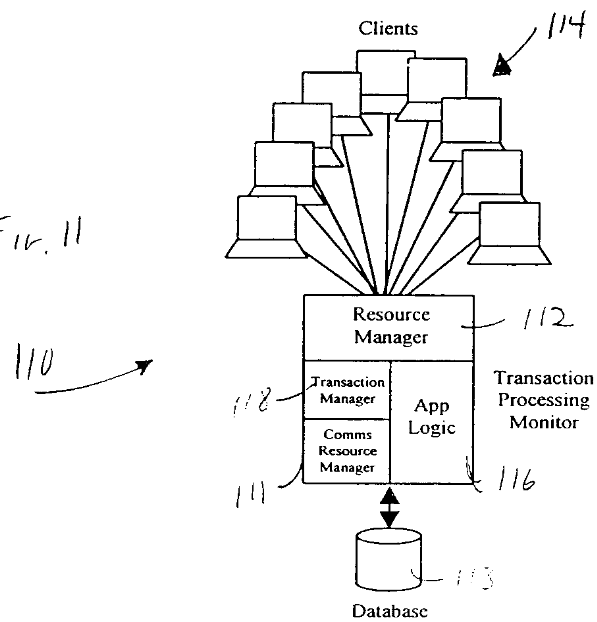


Fig. 11



124 122 126 123 128 123

Feature	Weight	MOM Product A Actual Score	MOM Product A Weighted Score	MOM Product B Actual Score	MOM Product B Weighted Score
Communication Styles					
Asynchronous					
Synchronous					
Point to Point					
One to Many					
Multicast					
Distribution List					
Dynamic Pub/Sub					
Message Delivery					
Once Only					
LIFO					
FIFO					
Priority					
Guaranteed					
Confirmation					
Routing					
Receipt Invocation					
Callbacks					
Polling					
Triggering					
Message Properties					
Translation					
Persistent					
Non-Persistent					
Message Type (Identifiers)					
Message Format					
Operations					
Configuration					
Management					
Location Transparency					
Environment (Portability/Scalability)					
Operating Systems					
Programming Languages					
Network Protocols					
Architecture					
Security					
Middleware Interop					
API					
Total Score:					

121

Technical Product Scorecard Matrix

129

120

FIG. 12

134 132 136 133 132 138 133

Non Technical Factors	Weight	MOM Product A		MOM Product B	
		Actual Score	Weighted Score	Actual Score	Weighted Score
Documentation					
Cost					
Ease of Installation					
Ease of Configuration					
Administration Costs					
Product Support					
Reputation of vendor					
Client skill base for product					
Client bias for vendor					
Total Score:					

131

139

130

Fig 13

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/41894

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 11/00, 9/44, 17/60, 17/00, 9/45, 17/30
US CL :714/25; 706/45; 705/7; 707/104; 717/1, 4, 5
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/50, 575, 704, 705; 705/7; 707/104; 717/1

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
WEST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,333,304 A (CHRISTENSEN et al.) 26 July 1994, the entire paper is relevant	1-26
Y	US 5,771,385 A (HARPER) 23 JUNE 1998, the entire paper is relevant	1-26
Y	US 5,956,513 A (McLain, Jr.) 21 September 1999, the entire paper is relevant	1-26
Y	US 5,745,880 A (STROTHMANN) 28 April 1998, the entire paper is relevant	1-26
Y	US 5,574,828 A (HAYWARD et al.) 12 November 1996, the entire paper is relevant	1-26

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
18 MARCH 2001	12 APR 2001

Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer Peggy Harrod THOMAS BLACK
Facsimile No. (703) 305-3230	Telephone No. (703) 305-9707

INTERNATIONAL SEARCH REPORT	International application No. PCT/US00/41894

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,761,674 A (ITO) 02 June 1998, the entire paper is relevant	1-26
Y, T	US 6,199,193 B1 (OYAGI et al.) 06 March 2001, the entire paper is relevant	1-26